



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/045,919	01/09/2002	Rajesh V. Patil	LOT920010027US1	3097

7590 07/20/2005
Shelley M. Beckstrand
314 Main Street
Owego, NY 13827-1616

EXAMINER

ALI, SYED J

ART UNIT PAPER NUMBER

2195

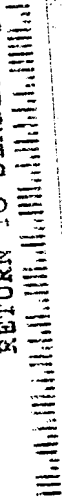
DATE MAILED: 07/20/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

RECEIVED
OIP/EPAP
JUL 28 2005

BECK314 138271004 1504 22 07/26/05
FORWARD TIME EXP RIN TO SEND
BECKSTRAND
61 GLENMONT RD
WOODLAWN VA 24381-1341

RETURN TO SENDER



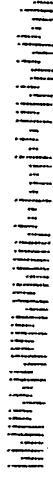
BEST AVAILABLE COPY

USPTO MAIL CENTER

JUL 28 2005

RECEIVED

UNITED STATES POSTAGE
000420479 JUL 19 2005
MAILED FROM ZIP CODE 22514
\$02.21
PENAL USE \$0.00
PRIVATE MAIL



A Better Windows

by Alexander Chang (written in 2002)

This is a list of suggested improvements for Windows XP and its built-in components. Readers are free to take these ideas for implementation.

This is the full article.

Extra improvements have been highlighted.

NEW SERVICES

Screensaver Hot Keys

-A hot key that (instantly) activates the screensaver. Why should I have to wait? Some programs will speed up when the computer is idle, therefore it will be wise to tell the computer exactly when you are away.

-A hot key (or an easily-accessed command) that disables the screensaver. This tool would be useful when watching movies, etc. (There should be an icon somewhere by the clock that reminds you to reactivate your screensaver).

Built-in Windows Script Writer

A hot key or a clickable icon that performs multiple tasks, activates different settings, orders tasks etc (all user defined). Note, if this idea is implemented, an abort command would also be necessary.

Examples:

Email Mode: Opens your email accounts in separate internet browsers (optional: types in passwords).

Maintenance Mode: Runs Scandisk, scans for viruses, defrags computer, and then closes.

Music Mode: Disables power-save, opens My Music folder, and launches your media player.

Boss Mode: Oh no, my boss is coming! Closes all non-work related programs, launches a Microsoft Excel file, places cursor somewhere.

Settings Mode: Sets the mouse sensitivity to your preferred setting.

Auto-Normalization Feature

A feature that will automatically normalize (and possibly amplify) all output sound. This will make the manual act of normalizing files unnecessary. It will also make manually adjusting the speaker volume between songs unnecessary.

A cool idea would be the use of the microphone and speakers to determine how high your speaker volume is set. This way, Windows can adjust the volume of various sound effects (your music will be loud, yet sudden unexpected chimes will not be).

Amplify Sound with Software

Sometimes the volume controls simply do not go high enough – especially when wearing headphones. This feature will allow the user to amplify sound even more. This feature should only amplify the sound of one particular file. It *should not* also amplify other noises such as sound effects (otherwise the user will get a heart attack).

Windows Alarm Clock

Wake up to your favorite mp3s or music videos. Have the weather report and your e-mail facing you right when you hit the snooze key. There are already programs that do this out there, but they do not work very well because of Windows incompatibilities such as standby, screensavers, and intermediate interruptions. An alarm clock built into Windows should fix all that. (The computer must be left on overnight.)

More control of uploads on network

Several things:

1. Notify (not with a pop-up, more like an icon appearing by the clock) when someone is downloading from your shared folder.
2. Display the downloader's statistics (IP address, which file(s) he is downloading, which files are queued, speed of download, duration of download, estimated time of download, etc). Also, keep these statistics in a history.
3. Restrictions Options: number of simultaneous downloads, maximum download speeds, disable streaming, blocking ip addresses, times of the day/week when you disallow any downloading, ability to cancel uploads, etc
4. Optional: be able to chat with downloader.

Quarantine for programs

A tool which can completely restrict specific programs from accessing the internet, popping up, using resources, etc. This is useful for preventing annoying programs (such as built-in ads, spyware, auto-update services) from even launching.

An alternative to quarantine would be the option of restricting a certain program's allowable usage of memory and CPU, as well as the time of day that it can run.

Detect if a Program has been repeatedly opened and closed by the user

Because closing and restarting consumes additional resources, this service will ask you if you would like to hide (instead of close) the program the next time you accidentally close it. Because this tool may become annoying under certain cases, it should contain a history as well as preference settings.

Fix the Notification Area (where the clock is)

Current bugs: The expand arrow sometimes disappears
Sometimes the program's icon still shows even after it has closed.

Improvements: Highlight icons on mouse-over

Contain an option menu that sets which icons should be on top, be on bottom, or not even show. This menu should be activated by right-clicking the icon in the Notification Area.

Note: icons should highlight only when the mouse is over the icon itself – thus leaving more free blue space for a right-click.

Windows Equalizer

How about a built-in equalizer in the operating system instead of just on media players? That way, all sound will be equalized.

Detect if a Program has been repeatedly opened and closed by the user

Because closing and restarting takes resources, this service will ask the user if he would like to hide (instead of close) the program the next time he accidentally close it. Because this tool may become annoying under certain cases, it should contain a history as well as preference settings.

Search for Identical Files

This service will search within folders or drives for identical files. This would be great tool for users who are not aware that they have similar files.

This tool should be able to handle several cases:

1. Exact same file, exact same filename
2. Exact same file, different filenames (not limited to "Copy of ...")
3. Cool but Optional: 2 files which are very similar

Example: Word documents that *start* with the same text, except one was discontinued.

Example: Movies (or songs) which contain a similar segment, thus redundant.

4. Obviously, system files should be ignored.

A service that detects and fixes slow performance (using Regedit)

Why should users have to tweak the Registry Keys themselves? Windows should be able to detect patterns.

Examples: Handling .avi's with explorer, Not Auto-running CDs, Faster internet browsing, etc

Abort Hot Key

A hot key which will abort a program that is in the process of starting up. This is useful for programs that take a while to start up (that were accidentally launched) and go through a lot of pop-ups before closing.

Another idea is an easy-to-use hot key that specifically disables or enables error reporting. If the user knows in advance that he does not want to send an error report, why should he wait for the error report pop-up to show? He can just hit the hot key.

This abort option should also apply to other annoying features such as auto-run CD, accidentally hitting shutdown or print, etc

Visual Speakers (for the hearing impaired, or for users who have muted their speakers)

When there is a sound effect, a visual indicator will appear at the corners of the screen (stereo). These indicators can be similar to the LEDs on old boom boxes. However, they can also be layered or color-coded to indicate frequency as well as gain. This feature should also work for games played in full screen.

Program Statistics

A background service that will log and record statistics on programs. Users can review which programs they use most, the resource usage of programs, whether programs run without their knowledge, etc.

IMPROVEMENTS

User can rearrange order of taskbar buttons

The buttons on your taskbar can be rearranged by dragging them around.

Fix icon/program mismatch

Current problem: File icons currently do not necessarily match the program that opens the file.

Solution: Fix it

Another problem: Some data files that have extensions used by, but unassociated with, certain programs are still launched by these programs. This generally happens when the user accidentally double clicks the file.

Solutions: realize that the file-type is wrong before launching the program. OR, pre-associate which files do not use the program.

Customize Desktop Icon Arrangement

Allow the user to *fix* his icon arrangements in *certain areas*. In other words, the desktop can be divided into sectors where the auto-arrange applies to just that sector. Likewise, the user can specify the area where downloaded files are stacked.

It also wouldn't hurt for multiple files dragged to desktop to be arranged nicely (instead of diagonally).

[Start Menu] Hide/Expand Option

This idea is similar to the current way Microsoft Office hides infrequently used buttons in menus. The Start Menu is often filled with many directories and programs that are rarely used. The user should be able to hide these. Example: Readme Files, Uninstall Files, etc

Deleting/Moving

Current problems: Sometimes users cannot delete/move files (because they are being used -- usually by Explorer).

Possible Solution: Instead of warning the user, Windows will automatically wait for the process to end and then perform delete/move action. This provides users with the assurance that their commands will return results.

Another Possible Solution: Smarter (or faster) use of Windows Explorer. Usually the reason files will not delete/move is because of a file lock. There is generally no need for these actions of Explorer when deleting/moving.

Another problem: When moving/cut-pasting directories, the source folder will still stay sometimes. Users cannot delete or open this folder.

Another problem: When moving multiple files and directories, the process will be interrupted if one of the files is in use. The problem is that other files that CAN be moved do not get moved, and the user is left with half of the selection at the source and half of the selection at the destination (a big mess when directories are involved).

Possible Solutions: (1) Do not let the move operation begin if one of the source files is running. (2) Give the user a choice at the interrupt pop-up to continue moving the rest of the files; then, detect the reason why the file cannot be moved. Use proper judgment after that: if the file can not be moved because it is being viewed (read only), then copy it to destination and delete it later.

[Replace File?] No to All Options

Three points:

1. Currently there is a No, Yes, and Yes to All option. There should be a "more options" tab that gives more options:

- **No to All:** Do not download and replace files that have filenames identical to ones that I already have

- **No but Rename:** Download this particular file but do not overwrite, just rename it. ex: filename(1)
- **No to All but Rename:** Same as above but to All

2. When the download is paused because it is awaiting the user's input, why not just keep downloading the rest of the files? Wouldn't it be more time efficient?

3. The user can *pre-define* his choices before they are prompted (ex: pre-define "no to all"). Under these circumstances, he should also get a download history where he is informed of which files were replaced, or renamed, etc.

Built-in Buffer Under-Run Prevention

This feature is currently only on CD-writing software during the burning process. Why not apply this to other things like playing movies? It should be activated either through a hot key or a few mouse clicks; after that, Windows will maintain the buffer.

[Task manager] : [Networking] displays I/O traffic, history, details

- The networking section should display a list of programs currently using the net. The program's traffic speed, net send/receive traffic, and traffic time should be displayed by the program.
- Users should be able to view a history of programs that have accessed the net (the log includes the specs above)
- *Optional:* The user can block programs from accessing the net

Smarter FIND tool for Internet Explorer (and Notepad)

The Find tool currently only features: whole word, case, direction (up down). It should have additional features:

Search all frames: for pages that use frames

Loop the search: keeps going after it reaches bottom/top

Change Highlight Color: for pages with strange background colors. User can see the highlight

Position Highlight: The highlighted word will not be positioned under the search window.

Very informative tutorials

In addition to Windows Help, why not have Windows tutorials? Very good ones include:

- A hot key tutorial: where the hotkeys are explained
- Tutorials for the user to add customized hotkeys, change hotkeys, and deactivate hotkeys
- A command tutorial: where the user learns about msconfig, regedit, ipconfig, etc
- Other tutorials for special features: Remote Desktop, setting up a LAN, etc

Smarter Disk Defragmenter

Disk defragmenting software should be improved as the modern era sends and receives more internet traffic.

Current Problems: To run properly, the defragmenter requires 15% free space on your hard drive. This is an un-resourceful required amount of space.

The defragmentation is not always successful (even with 15% free space)

Current Advantages: The defragmentation algorithm is simple.

The user can instantly stop Defrag without damaging files.

My Suggested Alternative:

Alex's Super Duper Disk Defragmenter

The user can specify many things. These include:

What type of defrag they want

1. Get rid of all the red (red = fragmented files)
2. Get rid of the white stripes (leaving just a big hunk of blue, big hunk of white, and of course green)
3. Combine green stripes into 1 chunk
4. Focus only on combining the zillions of small red stripes (files with thousands of small segments)

What lengths of extremity can the defrag program undergo:

1. Use *memory* to temporarily hold data (this is unsafe if the defragmentation program is suddenly halted)
2. Compress files (this takes a lot of time)
3. Purposely fragment whole files during the defragmenting process (sometimes more efficient).
5. Copy fragmented files to free space in secondary hard drives during the defragmenting process (improves efficiency)
4. Use the normal mode

Which files or sectors to focus on

The defrag program will put priority on defragmenting a particular file or section in the hard drive.

The user can specify how long he wants to run disk defrag. He can also inform the defragment program that he will leave the computer idle.

This information lets the defragmenter know which methods are the most appropriate.

Shutdown all unnecessary programs?

The user can order the disk defrag program to shut down all unnecessary programs while running.

More options.

The user can tell the defragmenter to shutdown, to re-launch programs, etc when it's done.

Repeat Disk Defrag if unsuccessful?

To be used if user only chooses a normal mode defrag

Perform Disk Defrag in DOS/Safe mode?

Done primarily to be faster, safer, and combine the green stripes

Other Improvements:

"Mouse in Corner" Idea

If the user leaves the mouse in the corner of the screen, it means that it's ok to use memory...

[Task manager] → [Processes tab] should contain descriptive details

In the form of a balloon on mouseover (or something else), the user should be informed about what each process does and what programs it is associated with

Example: the purpose of each running svhost.exe should be explained in general; and, the specific program that svhost.exe is running for should also be named

[INTERNET EXPLORER: History] more user friendly

The current configuration just shows a bunch of folders sorted by name. I propose adding more sorting options:

- let the user expand all of the folders at the same time
- let the user sort the folders/files by the *time*, *frequency*, or *duration* that they were accessed.
- let the user sort folders by the number of pages within the folder.
- let the user sort folders by the total size of files within the folder (size includes download cache from the site)
- let the user sort files (within folders) by priority - putting the main page first, the ads last
- truncate long page titles. *Also consider doing this for task bar buttons*

Ex: "Student Engineers' Council – Magazine" *becomes* "Magazine"

-SCARY: users can view the exact order in which user accessed/closed pages.

Note: The implementations of these new viewing options can be accomplished by using color-coding, hide/expand, indents, etc.

Disable INSERT key (or make it very obvious that it's on)

Add an easily-accessed command that will nullify the insert key. Or, change the look of the cursor (and/or mouse) when the insert key is on. You can also apply this to Caps Lock.

Hide Printers and Scheduled Tasks on Network

Currently there is no easy way of hiding the printers and scheduled tasks icon on your network. There should be an option to get rid of these.

Open With...

The Open With list currently appears after a long wait. The list also does not include a complete list of programs. Windows should make the list appear faster and include ALL programs. To avoid clutter, many of the programs listed will be hidden (but can be expanded).

More reliable and Editable File Details

Currently, the details about files (such as Date Modified, etc) are not always reliable. Fix this.

Users should also be able to change details (ex: change the Modified Date) or customize the way these details are calculated.

OTHER

Option to temporarily disable Start button during games

Add this option to the properties menu of the .exe file.

Option to play games (and run programs) in a resizable desktop window (instead of Full Screen)

Add this option to the properties menu of the exe file. This applies to old games as well as modern games

Sort thumbnails by their *hue*.

Why not make a neat option that sorts thumbnails by hue? Similar pictures or movies that have different names or sizes can be easily recognized this way.

Option that displays hardware information

Display voltage, temperature, CPU usage, memory usage, the time computer has been on, etc.

Option to communicate/complain with hardware

In some cases where the CD-rom or Hard drive is making too much noise, it would be nice to tell the computer to adjust the read speeds until the noise is reduced.

Transparent Windows

This option allows anything that uses a window to become transparent. The user can set the percentage of transparency. Another feature that may go well with this is an “Always on Top” option. This applies to the start menu, movies, and general popup messages.

Built in Anti-virus, Anti-spam, Anti-ad, Anti-popup, Intrusion-Alert, and incoming call notification.

Fairly self explanatory. It may not be the best, but it's there.

Snap Option

An option that allows windows to 'snap' together, or snap to the edges of the screen.

Better Mouse Software (under control panel)

The modern mouse has many new buttons and features. Window's mouse settings should allow the user to disable buttons, edit a button's affect. Especially for mouse wheel up/down, clicking the mouse wheel. These features should also work during games.

Detect unused files

A service that will find files that the user has left alone for a while. This is especially useful for files left by uninstalled programs. The service should use professional intuition – guiding users on what these unused files were for, what consequences there will be for removing them, etc.

Software-End Monitor Settings

Instead of using the monitor's options for Brightness, Contrast, Size, Position, Shape, etc, Windows should have an option under the control panel that adjusts the output being displayed.

GroupBar: The TaskBar Evolved

Greg Smith, Patrick Baudisch, George Robertson,
Mary Czerwinski, Brian Meyers, Daniel Robbins, and Donna Andrews
Microsoft Research
{gregsmi; baudisch; ggr; marycz; brianme; dcr}@microsoft.com

Abstract

Our studies have shown that as displays become larger, users leave more windows open for easy multitasking. A larger number of windows, however, may increase the time that users spend arranging and switching between tasks. We introduce GroupBar, a task management system for dealing with the profusion of windows on the PC desktop. Designed to offer the same basic form and function as the existing Microsoft Windows™ TaskBar, GroupBar additionally allows users to group windows into higher-level tasks and enables task switching with a single mouse click. In order to gain experience with GroupBar usage and to develop reasonable task definitions we conducted a longitudinal field study. Based on the results of that field study, we conducted a comparative user study wherein we found that participants were able to multitask faster when using GroupBar than when using the existing Windows TaskBar.

1. Introduction

Twenty years ago, Bannon et al. (1983) observed that information workers often switch between concurrent tasks. In Rooms, Card and Henderson (1987) observed that tasks can be supported by managing working sets of windows, in much the same way operating systems manage working sets in memory. They identified desirable properties of task management systems, including: fast task switching, fast task resumption, and easy reacquisition of mental task context.

Since then, many virtual desktop managers have been built and each exhibits some of these properties. Task management systems typically provide some efficient way of switching from one set of windows and applications to another set, as a basic form of task switching.

Although workers may switch among tasks in a self-guided manner, a significant portion of task switching is caused by external interruptions. Czerwinski, Cutrell, and Horvitz (Cutrell, 2001; Czerwinski, 2000; Czerwinski, 2000b) have sought to understand the influence of interruptions on task switching for information workers

in order to design user interface tools that can assist users to recover from interruptions.

We have also been motivated to re-examine task switching and task management design opportunities in the face of the growing popularity of larger display and multiple monitor configurations. In an informal study at our corporation, we found that when users shift to larger display surfaces, they leave more applications running and associated windows open. For example, we observed that single display users tend to keep an average of 4 windows open at once, while dual monitor users keep 12 and triple monitor users keep 18 windows open on average ($N=16$ users). Although a larger study is required for verification of these results, this significant trend suggests that there is an opportunity for design innovation with windows and task management to make dealing with larger numbers of concurrent windows a fundamentally more natural and effective experience.

We have developed a tool to exploit this opportunity. GroupBar is a desktop-resident toolbar, similar to the Microsoft Windows TaskBar, which allows users to arrange windows into groups and to switch between tasks with a single mouse click. The similarity to the Windows TaskBar was chosen as a design point in recognition of the fact that we could leverage user familiarity to reduce learning time, as well as to provide a basis for targeted comparison of task management features we developed.

In this paper, we will discuss related work, describe GroupBar, and present the results of a longitudinal field study of GroupBar, and a comparative user study of GroupBar and TaskBar.

2. Related work: task management

The most popular software system for task management is the virtual desktop manager. One of the earliest designs exploring a virtual desktop manager was Smalltalk Project Views (Goldberg, 1983). Rooms (Card, 1987; Henderson, 1987) is probably the most well-known of these kinds of systems. A number of these systems are currently available, and are described in (XDesk, 2003). We found that none of these systems have been evaluated in a stringent way; we could not find detailed reports on

studies demonstrating how easy to learn they are, or how well they integrate into real-world settings.

In addition to virtual desktop managers, a number of novel solutions have been proposed, including extending the user's desktop with additional low-resolution screen space (Baudisch, 2001), 3D solutions like the TaskGallery (Robertson, 2000) and Wurnig (1998), zoomable space as in Pad++ (Bederson, 1994), and the use of time as the main axis (Rekimoto, 1999). Also, tiled window managers (Bly, 1986; Teitelman, 1986) address some of these same issues, as well as systems that involve bumping other windows away (Bell, 2000; Kandogan, 1997).

We have pursued prototypes of temporal and spatial visualizations of users' daily computing configurations. These designs use lightweight, temporal cues, such as the state of a user's desktop at different times (Malone, 1983). We are also building support for task-based visualizations and switching, in a similar vein to the work of Henderson & Card (1987), Kaptelinin (2002), Macintyre et al. (2001) and Robertson et al. (2000).

In distinction to the prior work, we have sought designs for the virtual desktop organizers that don't replace the entire PC desktop with a new metaphor, but rather occupy the same conceptual and physical space as is already devoted to window management in the Windows OS – namely, the area along the edges of the display surface using the same minimized representations of top level windows. Using these prototypes, we are performing longitudinal studies on the benefits of temporal and visual cues for enhancing memory about knowledge-based tasks. We seek to understand the potential benefits from the use of these systems, and to iterate their design. For example, the Windows XP TaskBar provides “grouping by application” (executable) to address the problem of running out of space inside the bar. Grouping this way rather than by task has created confusion for many users, as specific application windows may not be related to each other, and cross-application windows might be related to the same task.

The ProjectBar, shown in Figure 1, was a first attempt to help users organize their documents and email by task (or “Project”), as opposed to by application. The default “Desktop” Project and three other user-defined Projects are shown. The currently selected Project, “CHI Paper”, is expanded, so it shows its contents in the bar – i.e. the five window tiles. The other, inactive Projects are represented by clickable Project buttons. Hovering the mouse over the non-selected Project, “eMail”, reveals a thumbnail fly-out image of what its desktop looks like.

The ProjectBar organizes open windows and applications so that associated items are grouped into a single Project. Projects are named and appear as a list of Project buttons in the bar, distinguished from the individual window buttons by color and shape. The list of Projects in the ProjectBar can provide a visual reminder of ongoing work and help with task management. When the user selects a Project by clicking on its button, the individual windows belonging to that Project are revealed on the

desktop and the associated window buttons are revealed in the bar. In Figure 1, clicking on the “eMail[3]” button would hide the five window tiles belonging to the “CHI Paper” Project and reveal the three tiles belonging to the “eMail” Project. When users return to a Project, windows re-appear as they had been laid out previously on the desktop, preserving spatial context for the user as well as the resources needed for continuing the task.

We performed a pilot user study comparing ProjectBar with the Windows XP TaskBar to examine the ProjectBar's ease of use for multitasking with knowledge work. ProjectBar was found to provide higher satisfaction than the TaskBar.

Despite these encouraging results, we were ultimately not satisfied with how much user effort was required to learn to use the ProjectBar. Like many other systems, the ProjectBar requires users to explicitly create a task or virtual desktop before windows can be assigned to it. An alternative, offered on some virtual desktop switchers, is a fixed set of empty, pre-configured tasks or desktops that can be populated. The creation of new custom-tailored tasks remains a bottleneck.

We therefore built a lighter-weight version of the ProjectBar, which we call GroupBar. Our goal in designing GroupBar was to lower the bar to task creation and maintenance by allowing users to create or remove a task on the fly using a single dragging gesture.

The other issue GroupBar addresses is the accessibility of windows belonging to different tasks. While virtual desktop managers tend to have a strict separation between tasks, GroupBar allows users to simultaneously display any subset of windows, even if they should be assigned to different tasks.

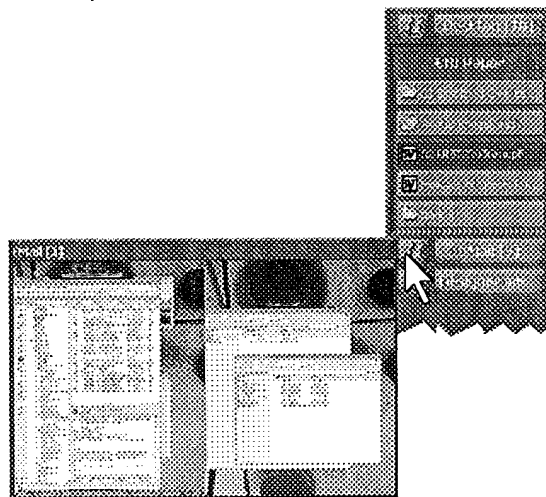


Figure 1: ProjectBar with four projects. As the user hovers the mouse over one of the projects, a fly-out provides a preview of that project.

3. GroupBar walkthrough

In this section we will describe GroupBar, how it allows users to interactively rearrange tiles representing windows, how it allows users to organize window tiles into high-level tasks called 'Groups', and how GroupBar allows users to switch between these tasks with the mouse.

Figure 2 shows two pictures of GroupBar, here in its vertical form factor. As in the Windows TaskBar, GroupBar contains one tile for each open window in the system. The current 'active' window tile is shown in a darker, depressed-button state, and any tile can be clicked on to activate (or to minimize, if already active) the corresponding desktop window.

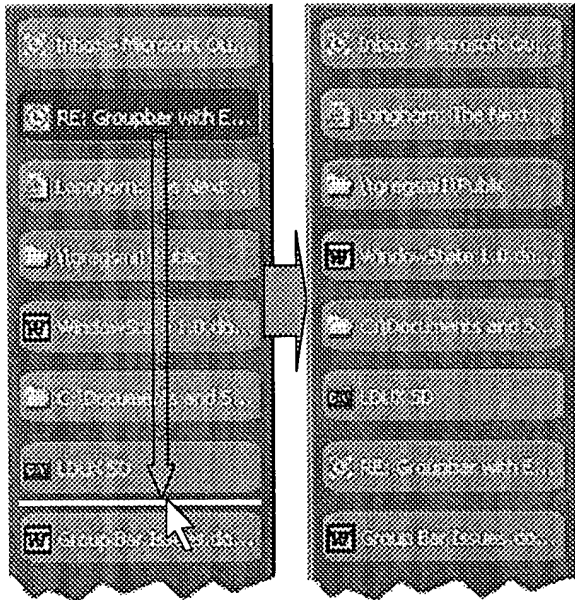


Figure 2: GroupBar allows users to rearrange tiles by dragging a tile between two other tiles.

The key enabling feature of GroupBar is the addition of drag-and-drop functionality to tiles on the bar. As the user drags a window tile along the bar, a white insertion caret appears, moving along the bar with the mouse cursor to indicate which of the possible drop locations will be activated when the tile is released (see also Figure 7).

The drag-and-drop functionality allows for two types of interactions. First, it allows users to *arrange* their window tiles freely, which may be used for giving tiles a more meaningful order or to establish a better correspondence between the position of a tile and the position of the represented window on the screen. Figure 2 shows the straight caret symbol that GroupBar uses to indicate that the dragged tile will be inserted at this location.

Second, users can drag a window tile *onto* another tile, which combines these two tiles into a Group. As shown in Figure 3, during the drag operation, a curved white caret (as distinct from the straight-line caret for

rearranging) is shown to indicate that the drop operation will result in Group formation. When a Group is formed, GroupBar visually unifies member tiles by surrounding them with a gray background and complementing the newly formed unit with a green "tab" at the top. Users can remove a tile by dragging that tile out of the group. Groups reduced to a single tile are automatically terminated and the group tab disappears.

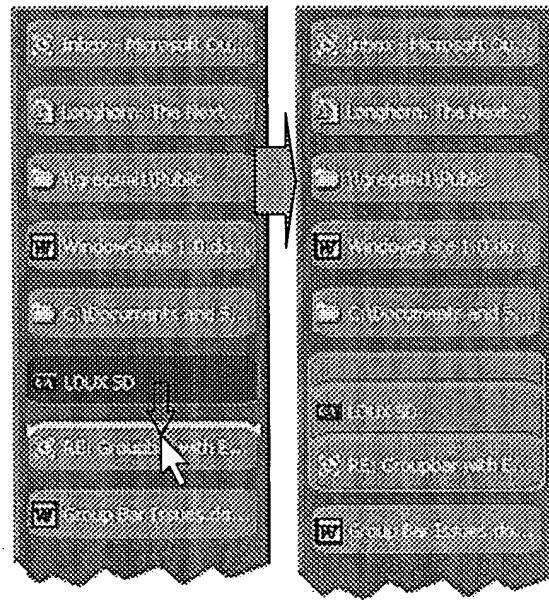


Figure 3: Dragging a window tile onto another tile combines both tiles into a "group".

The main point of GroupBar is that it allows users to perform operations on all of a Group's constituent windows simultaneously by manipulations of the Group tab. As further described in Section 4, the functionality GroupBar offers for Groups is directly derived from the functionality Windows offers for a single window tile. *Clicking* a Group tab restores all of the windows in the Group and brings them to the foreground. *Right-clicking* the tab offers a context menu for additional Group actions (see Figure 4).

While the above figures depict a vertical bar style, GroupBar can be configured to either horizontal or vertical form factors, just like the standard Windows TaskBar (Figure 5). And like the standard TaskBar, GroupBar can be transformed from one form factor into the other anytime by dragging the bar to a different edge of the screen.

As we will show in the comparative user study below, GroupBar's ability to allow users to create, modify, and interact with entire groups of windows helps them increase their window management efficiency while requiring only a modest learning effort. Before we present our two user studies, however, we will discuss four of the design points in more detail.

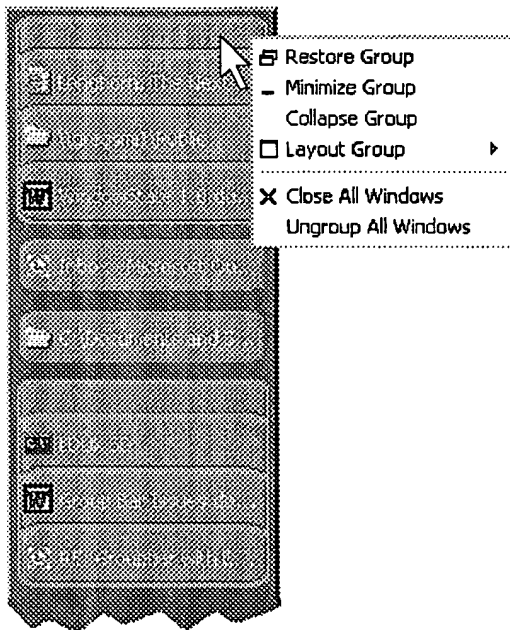


Figure 4: Clicking the green Group tab restores all of the windows in that Group and brings them to the foreground. Right-clicking a Group tab offers additional Group actions.

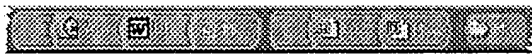


Figure 5: Fragment of GroupBar in its horizontal form factor.

4. GroupBar design

4.1. GroupBar is backwards compatible

The key addition GroupBar makes to the root interaction functionality of the existing Windows TaskBar is a drag interaction on the bar tiles. Since no drag interaction is defined on the tiles of the original TaskBar, users who choose not to use the grouping functionality can use GroupBar as if it were the regular TaskBar. When grouping is used, the additional grouping information is primarily conveyed via subtle changes in the shape and surrounding coloration of grouped window tiles, using the green Group tab as a high-contrast unifier. The size and visual distraction level of the Group tab is kept minimal by representing only the primary “switch to this Group” function directly in the interface; all other functions are relegated to a context menu. (Figure 6 shows an earlier design with an additional on-tab embedded button.)

One price to be paid for GroupBar’s backward compatibility and unobtrusiveness to users who do not need grouping capabilities is that the grouping feature may

have low discoverability. We plan to address this issue by using two different approaches. One is to have bubble help appear in situations where grouping functionality might be useful, for example after a user has performed a series of different window minimization/restoration actions. The other possibility is to explore algorithms for automatic grouping based on analysis of the user’s ongoing windowing activity, for example making an automatic Group out of a set of windows that seem to be highly co-incident on the screen. Since such automation runs the risk of being intrusive, it would have to be configured carefully. This feature was requested during our longitudinal field study.



Figure 6: Earlier GroupBar design: Without the tile curvature, the grouping is less apparent. This design features an additional “collapse” button for each Group.

4.2. Overloading drag interactions

GroupBar’s drag interaction was derived from the drag interactions found in Windows file management and the toolbar customization functions of Microsoft Office. Porting the drag interaction was generally straightforward with a few special adaptations. Since a dragged window tile can be both appended to a tile or Group, and also inserted between tiles or Groups, the GroupBar drag and drop design had to simultaneously support grouping and re-ordering semantics.

As mentioned in the walkthrough section, GroupBar accomplished this by distinguishing dropping into Groups from dropping outside of a Group by altering the shape and location of the caret. Namely, Group insertion carets are curved and placed up against the tile with which grouping will occur, while non-Group rearrangement carets are straight and centered in the gap between two tiles. Figure 7 shows how the different caret shapes visually distinguish the different operations even though drop targets are closely co-located. Enabling these multiple carets was one of the main reasons for using the curved window tile design (compare to the non-curved design shown in Figure 6).

To help users acquire the individual drop target areas, we chose to decouple the target surfaces from the visual location of the caret symbols. As Figure 7 shows, the screen space containing the straight-line caret is too small to allow users to acquire this space efficiently. GroupBar solves this problem by distributing the screen space between tile centers evenly among the three adjacent targets independent of where the caret symbols appear. Our first experience with this assignment of target surfaces is positive and the benefit of larger minimum

drop areas seems to outweigh the lack of absolute positional precision.

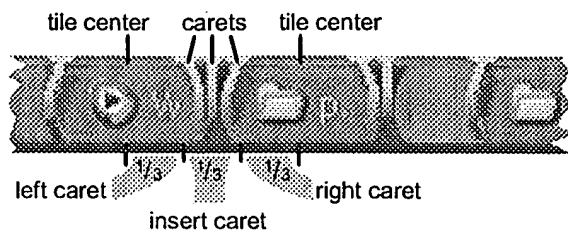


Figure 7: Bar fragment and all possible carets over it. In order to allow users to easily acquire the drop position for insertion, GroupBar distributes the activation surface evenly over the various possibilities.

4.3. Maximizing groups: window layout

The philosophy behind GroupBar is to allow users to operate on groups of windows as though they were a single unit. In existing Windows design, the left-click behavior of a tile button changes depending on the state of the window. For example, when a window is in a minimized state “Restore” is the most likely option, and “Minimize” is most likely when already restored. This is designed to optimize the button’s use and allow invocation of several different functions serially through a single control surface. For familiarity and efficiency, we brought the analogous facility forward into the Group tab: If the Group is all minimized, left-clicking the Group tab restores all the windows and brings them forward. If the Group is already all restored, left-clicking the Group tab minimizes all the windows.

One operation that does not translate immediately from individual window tiles to Groups is the “Maximize” function. Maximizing each individual window in a Group would make them all overlap one another, which is likely to be more problematic than useful. GroupBar therefore extends the analogy with a twist by creating a “Layout Group” operation which serves to maximize the *collective* space taken up by the Group, rather than maximizing the space taken up by any individual window. Since there are several ways to accomplish this type of cooperative maximization, GroupBar currently allows users to choose from a selection of predefined layouts (Figure 8), but we recognize that users might like to define their own spatial layouts.

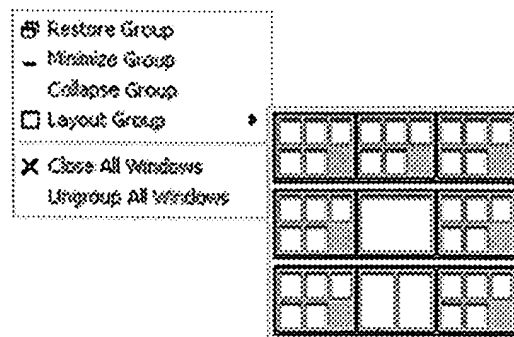


Figure 8: GroupBar context menu allows users to arrange all windows in that Group according to predefined layouts. Here the user uses a triple-monitor display, thus every layout extends across three screens.

4.4. Handling large numbers of windows

As the number of displayed windows increases, any type of bar interface will eventually run out of space. The original Windows TaskBar deals with the issue by making users page through sets of tiles using small arrow handles (Figure 9a). Since this approach makes a large number of potentially relevant tiles difficult to access, GroupBar implements a different approach that leverages its knowledge about Groups. If GroupBar runs out of space, Groups can collapse into a more space-efficient representation in order to make room and prevent the bar from overflowing (Figures 9b and 9c). Groups to be collapsed are picked on a least-recently-used basis. We believe this will almost always be preferable over the “agglomeration by application” strategy implemented in the current version of Windows.

In order to allow users to use collapsing functionality to reduce visual clutter even before the bar overflows, users can manually collapse and expand Groups from the Group context menu.

As an alternative strategy, GroupBar users can deal with large numbers of windows by creating additional bar instances. Additional bars are added using the “Add New Bar” command from the GroupBar context menu. New bars are created empty. The user can position the newly created bar on any edge of any monitor, and then populate it using the already described drag-and-drop mechanism. By allowing users to place window tiles and Groups at widely separated screen areas, additional bars allow users to leverage spatial memory more heavily.

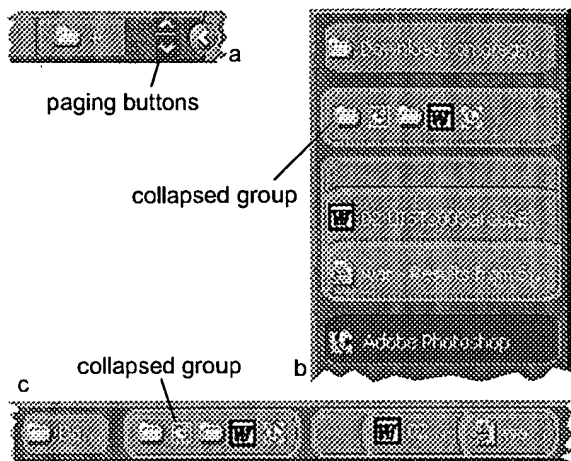


Figure 9: (a) Taskbar overflow vs. (b, c) Collapsed Groups in GroupBar.

5. Longitudinal user study

In order to gather information concerning how people actually use virtual desktop managers, and to begin to understand specifically how GroupBar might be used in real situations, we performed a longitudinal field study on a small number of subjects over a 7-10 day period.

5.1. Method

Five participants, aged between 20 and 60 and all male, were recruited to participate in the study. The participants were screened from a large local database of volunteers based on the criteria that they were experienced Windows and Office users. The users were also recruited to match design "personas" developed to represent target user groups. Space does not permit including the detailed specifications of each persona, but, broadly, the three personas used in the study included: 1 analyst, knowledge worker and light developer, 2 consultants, both knowledge workers, and 2 IT professionals. As may be obvious from the descriptions, all participants were fairly technology savvy.

A field study methodology was utilized in order to examine the usefulness and usability of GroupBar compared to the existing TaskBar. We used an *in situ* method to study the use of dual monitors and GroupBar in order to establish how important the new GroupBar features were for multiple monitors *using the participants' own work information and habits*. If, after approximately one week of use, participants were using the grouping features of GroupBar with their larger display space, this would provide evidence of the system's usefulness.

The five participants were first visited in order to collect baseline measures before starting the study proper. The baseline metrics included a measurement of how many windows users kept open with the TaskBar, inter-

view details about how they used their TaskBar and monitor for their work, and a survey about TaskBar features and relevant patterns of window management behaviors.

After the baseline visit, the 5 users were provided with the GroupBar executable and a very brief email tutorial on how to use its grouping features. Participants agreed to try GroupBar out as their main taskbar for 1 week. After using the new bar, they were visited in order to collect details and comments about how they utilized GroupBar, and how useful they found it to be. We also interviewed them about the kinds of tasks they chose to leverage GroupBar for, with an eye toward informing our design of the lab study, also presented in this paper, using realistic tasks.

5.2. Results

5.2.1. Baseline survey and usage data

Before using GroupBar, the participants reported typically running 5.6 applications or programs at once, on average, and that they kept an average of 6.4 windows open at once. All users reported using the original TaskBar in the horizontal position. Two of the participants used Auto-hide, Always-on-top, Group-by-application and the "Close All Windows" feature available when Group-by-application is turned on. Only one participant reported using the "Tile windows vertically/horizontally" function in the TaskBar, and no one reported using the "Cascade windows" function. Only one of the participants knew that right-clicking on a window tile allowed them to close that individual window. This background data shows a fairly standard usage of the TaskBar, with users typically only using the minimize/maximize and close capabilities related to the TaskBar tiles.

5.2.2. Usage and user satisfaction with GroupBar

Once it was installed, all users were able to easily learn how to use the grouping features in GroupBar. We received no emails or calls for assistance after sending the installation instructions. In addition, users were able to easily integrate it into their existing work practices, as evidenced by their comments and grouping habits. This is what we had hoped to observe, and was our initial design goal.

After using GroupBar for one week, four of the participants filled out a user satisfaction questionnaire about the perceived benefits of the system, and areas in need of improvement. (One user did not return his questionnaires.) Users on average organized 2.5 groups within GroupBar, and 2 windows in each group, on average. Three of the participants chose to run GroupBar vertically, and 1 ran it horizontally. The user satisfaction findings were reasonably favorable overall for a first iteration user study, and two of the participants stated they

would continue to use GroupBar after the study, despite its lack of integration with the real TaskBar. GroupBar scored above average in response to questions such as, "It is useful to be able to group the tiles on GroupBar by dragging them 'on top' of each other", "It is useful to be able to close/open a group of windows all at once", "It is useful to have GroupBar remember a layout for a group of windows, so that they open in the same layout as when I closed the group", and "GroupBar makes multiple monitors more useful". Users were not as positive about having non-group windows minimize on a group switch, or about running more than one GroupBar at a time. All of the average ratings are listed in Table 1.

Question	Average Response (1=Disagree, 5=Agree)
It is useful to be able to rearrange the tiles on GroupBar by dragging them into a different order	3.5
It is useful to be able to group the tiles on GroupBar by dragging them "on top" of each other.	3.75
It is useful to be able to open a group of windows all at once.	3.5
It is useful to have GroupBar remember a layout for a group of windows, so that they open in the same layout as when I closed the group.	3.5
It is useful to be able to close a group of windows all at once.	3.75
It is useful to have windows that are not in use minimize when I click on another group ("Minimize on Group Switch").	2.3
It is useful to be able to have more than one GroupBar running at a time.	2.0
I spend less time re-laying and re-sizing windows now that I have GroupBar.	3.0
GroupBar makes multiple monitors more useful.	3.25
GroupBar makes multiple monitors more pleasant to use.	3.0
I feel that the two monitors with the Group Bar provide me with enough desktop space.	2.75

Table 1. Average user satisfaction ratings for GroupBar.

We asked users what features of GroupBar most helped them manage their open windows. Here were the responses from the four participants that provided comments:

- Participant 1: Being able to open 3 windows for different features of Outlook as a group has been very helpful. Still trying to get used to having multiple internet windows open.
- Participant 2: Used it much like I use the task bar to select the window I want to use.
- Participant 3: Being able to open, and close, a group of windows at the same time. When I restore a grouped set of windows, it remembers the positions of the windows.
- Participant 4: I can think of at least 3 sets of groups I would like to be able to launch on a single click. When they do come up, I want them grouped already.

Next we asked what more we could do to design better windows/task management support into GroupBar. Suggestions from the same four participants that provided comments are included below:

- Participant 1: It would be nice to have the start button and shortcuts on the group bar and eliminate the Taskbar altogether.
- Participant 2: It would also be nice to be able to right click close a window from GroupBar. (*Note: We have since added this capability into GroupBar.*)
- Participant 3: When a new window spawns from a grouped application, group it with the others. For example, when I open an email from Outlook it doesn't group with Outlook. I have the same issue with IE. A new browser window does not group with IE. I don't like having two bars (the TaskBar and GroupBar). Combine the capability of the Group bar into the Task Bar. I would like to stretch the TaskBar across the bottom of both monitors.
- Participant 4: Give GroupBar the smarts to auto-group icons that are commonly grouped together on each person's desktop. Allow us to pre-configure sets of applications that should be grouped together by default. I would like to be able to click an icon on GroupBar (or whatever) and have it launch a set of grouped applications specific to a task or function.

5.3. Discussion

The user study provided initial evidence that some of GroupBar design decisions were deemed valuable by users. For a first iteration design, users were able to easily figure out how to use GroupBar and were using the grouping features for their common tasks. Given the lack of ethnographic data available to designers of task management systems, this is an encouraging finding. In addition, two users stated they were going to continue using GroupBar after the study. These same users also liked the fact that the windows' positions were preserved in

GroupBar, so this feature will be maintained in future iterations.

The study further indicated interesting design ideas for improving GroupBar. First, we knew that GroupBar should be integrated with the current TaskBar but we did not take the time to add the Start Menu, system tray, etc. to the initial prototype. Users clearly wanted this and future work will head in this direction. Second, one user requested that children windows should group with the parent window's group. Finally, at least one user suggested that we move in the direction of allowing GroupBar to "auto-group" applications that are typically joined together, based on frequency or recency of use.

We decided to take the initial feedback from our small longitudinal study and gain a more robust understanding of GroupBar's hypothesized ease of use over the TaskBar by performing a laboratory study. The tasks we observed users performing with GroupBar in the field informed our choice of tasks to utilize as part of the lab study's design. In addition, we included a much larger sample of users. We expected to see significant performance benefits when multitasking across various documents and applications when using GroupBar, as opposed to the TaskBar.

6. Comparative user study

In our laboratory investigation, we created 3 tasks consisting of between 2 and 3 documents each, matching what we saw on average in the field. The tasks consisted of a "Spreadsheet", a "Joke" and an "Image" task. The Spreadsheet task required participants to go to selected cells in an Excel spreadsheet as indicated by a Word document, copy the contents of the cell at that location (a 9 digit random number), and paste it both in the Word document and in another Excel spreadsheet. The Joke task required users to identify typographical errors in a list of jokes in a Word document, copy them and paste them and the page number on which they occurred in an Excel spreadsheet. The Image task required participants to modify images in PowerPoint and in Paint based on instructions in a Word document. Two isomorphic sets of each of these tasks were devised so that they were of approximately equal difficulty (e.g., the random numbers in the Excel spreadsheet were rearranged, as were the selected cells, both Joke documents consisted of 2300 characters and had 23 typographical errors but were different jokes, and the instructions for how to modify each image simply asked the user to use different simple shapes or different colors).

6.1. Method

Eighteen participants (half female), all multiple monitor users and very experienced MS Office users as identified by a validated screener, were recruited for this study. Participants were 34 years old, on average, had used the

computer for an average of 14 years, and said they multitasked between an average of 6 tasks.

Participants were given instructions about the overall study procedure and then allowed to read a brief overview of how each tool, the TaskBar or GroupBar, worked before proceeding. All participants were very familiar with the TaskBar and knew of most of its features, even if they didn't choose to use them. In order to ensure that they learned how to use GroupBar, users were guided through the grouping and layout of the documents that formed each of the three tasks for the study. In the TaskBar, participants could arrange the items in the TaskBar by application (as supported in the software) but not by task. However, we did allow them to lay out and size their task windows in a way that was best suited to each task before beginning. The TaskBar and GroupBar were both laid out vertically along the left-most bezel of the left-most monitor. While running GroupBar, the TaskBar was laid out horizontally and put on "auto-hide", so that users never saw it and interacted solely with GroupBar. Finally, the "agglomeration by application" mechanism in the TaskBar was turned on so that window tiles of like applications were juxtaposed, but they did not collapse into a single tile menu of all application windows.

In order to ensure that participants had to task switch between tasks, the experimenter interrupted them at set places in their tasks. Five task switches were required in order to carry out all three tasks to completion, the first three of these guided by experimenter interruptions between the Spreadsheet task and the Joke task when participants were approximately $\frac{1}{4}$ of the way through each, and then again at approximately $\frac{3}{4}$ of the way through the first task (Spreadsheet). After the first 3 interruptions, participants were told to finish the Joke task that they were in, go back and finish the Spreadsheet task, and then switch to the Image task and carry it through to completion. A twenty minute deadline procedure was used for each of the three tasks to keep the session length under two hours.

The study was run on two identical, late model Compaq Evo machines with triple flat panel LCD monitors running at 3840 x 1024 resolution. Late model MS keyboards and the IntelliMouse were used for input. Windows and Office XP were the base OS and applications used in the study. The order of software tool used and task set were counterbalanced across participants. Participants were run in pairs each session.

Dependent measures collected included task time, subjective satisfaction responses to a questionnaire presented after using each tool, and overall tool preference. Task times were recorded using a countdown program on the participants' machines. A log of users' activities in terms of window management and group interaction was collected; analysis of that data is ongoing.

6.2. Results

The task times revealed a strong positive skewing, therefore a log transformation was applied as is standard to correct for a non-uniform response time distribution. A t-test of the log task times revealed a borderline significant task advantage for GroupBar, $t(34)=1.5$, $p=.07$, one-tailed. GroupBar average task time was 11.7 minutes, while the average for the TaskBar was 13.25. The task time data including one standard error of the mean in each direction are presented in Figure 9.

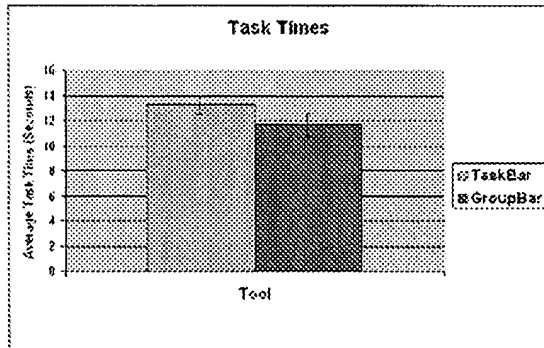


Figure 9: Average task times +/- one standard error of the mean for TaskBar and GroupBar.

Survey Question (1=Disagree, 5=Agree)	TaskBar	GroupBar
Task switching was easy to perform using the...	2.95	4.63
It was hard to go back and forth between my various windows and applications using.....	3.32	1.42
I was satisfied with the functionality of the	2.68	4.42
The TaskBar/GroupBar is an attractive innovation for Windows.	3.16	4.42

Table 1: Average satisfaction ratings for the TaskBar and GroupBar. All ratings were significantly in favour of GroupBar at the $p<.05$ level.

In terms of overall satisfaction with the software, participants strongly favored GroupBar over the TaskBar. Individual t-tests of each questionnaire item revealed satisfaction ratings significantly in favor of GroupBar on every question (an overall ANOVA was not possible as questionnaire items were worded both in the positive and in the negative in an effort to avoid presentation bias).

Strict confidence levels were adopted (Bonferroni corrections) to account for multiple tests but all p-values were well below the .001 level. Table 1 summarizes the average ratings for each question.

Finally, GroupBar was unanimously preferred over the TaskBar. Despite this, many participants suggested improvements to GroupBar. Most frequent requests were for color coding or labeling of the groups in the bar, and for tooltips for document names when the group is collapsed. These features can be easily added to GroupBar. Several expert users wanted to see better keyboard accelerators like ALT + TAB enabled in GroupBar as well.

6.3. Discussion

While the field study suggested to us that GroupBar was discoverable and considered valuable by the participants, the laboratory study allowed us to better verify these benefits in a more controlled setting. Using tasks similar to what we observed users doing in the field, we had users switch back and forth between multiple tasks, prompted by interruptions or task completions. GroupBar provided benefits over the TaskBar both in terms of overall task time (a borderline significant result) and in terms of users' perceived satisfaction with the task switching (strongly significant). Users commented that the tasks and interruptions forcing the switches were similar to what they experienced in the real world, so we feel we succeeded in simulating an information worker's daily task juggling. The study provides further evidence that software tools like GroupBar can provide user assistance as users manage multiple, complex tasks.

7. Conclusions

GroupBar provides basic task management, making it easy to group windows into a task using a straightforward drag and drop interaction. Task switching is accomplished with a single mouse click. Additional functionality on all windows in a task (e.g., minimizing, restoring, and closing) has been determined through two studies to be easy to use.

The major difference between GroupBar and other task management systems lies in how the tasks are shown to the user. Rooms and other virtual desktop systems show all the tasks only when users go to an overview. GroupBar displays the tasks in a form that is a subtle extension to the familiar Windows TaskBar. This was shown to be a great advantage in terms of learnability and user acceptance, as demonstrated by our *in situ* and laboratory user studies of GroupBar.

Further user studies of GroupBar and its extensions will lead to design improvements and help clarify which designs are most appropriate across users and various display configurations. Competitive studies will clarify advantages and disadvantages of these two systems compared with existing systems. It is clear from our early studies on these topics that such new task management

systems are considered valuable by information workers, especially in multiple monitor configurations.

8. Acknowledgements

The authors would like to acknowledge the contributions of John Pruitt for help in designing the longitudinal user study, and Eric Horvitz for comments on early drafts of this paper.

9. References

- Baerbak Christensen, H. (1997). *A software development environment based on a geographic space metaphor*. Technical report, Univ. of Aarhus.
- Bannon, L., Cypher, A., Greenspan, S., and Monty, M. (1983). Evaluation and analysis of user's activity organization". In *Proc. CHI'83* (pp. 54-57). NY: ACM.
- Baudisch, P., Good, N., Stewart, P. (2001). Focus plus context screens: combining display technology with visualization techniques. In *Proc UIST 2001* (pp. 31-40).
- Bederson, B. & Hollan, J. (1994). Pad++: A zooming graphical interface for exploring alternative interface physics. In *Proc. UIST'94* (pp. 17-26).
- Bell, B. and Feiner, S. (2000). Dynamic space management for user interfaces. In *Proc. UIST'00*, (pp. 238-248).
- Bly, S., Rosenberg, J. (1986). A comparison of tiled and overlapping windows. In *Proc. CHI '86* (pp. 101-106).
- Bury, K. F., Davies, S. E., and Darnell, M. J. (1985). Window management: A review of issues and some results from user testing. *IBM Human Factors Center Report HFC-53*, San Jose, CA.
- Card, S.K. & Henderson, A.H. Jr. (1987). A multiple, virtual-workspace interface to support user task switching. In *Proc. CHI+GI 1987* (pp. 53-59). NY: ACM.
- Cutrell, E., Czerwinski, M. & Horvitz, E. (2001). Notification, Disruption and Memory: Effects of Messaging Interruptions on Memory and Performance. In *Human-Computer Interaction--Interact '01* (pp. 263-269). IOS Press.
- Czerwinski, M., Cutrell, E. & Horvitz, E. (2000). Instant Messaging and Interruption: Influence of Task Type on Performance. In *Proc. OZCHI 2000* (pp. 356-361). Sydney, Australia.
- Czerwinski, M., Cutrell, E. & Horvitz, E. (2000b). Instant Messaging: Effects of Relevance and Time. *Proc. HCI 2000, Vol. 2*, (pp. 71-76). British Computer Society.
- Czerwinski, M. & Horvitz, E. (2002). Memory for Daily Computing Events. In *Proc. HCI 2002*, (pp. 230-245).
- Goldberg, A. (1983). *Smalltalk-80*. NY: Addison-Wesley.
- Grudin, J. (2001). Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proc. CHI'01*, (pp. 458-465).
- Guimbretiere, F., Stone, M., and Winograd, T. (2001). Fluid interaction with high-resolution wall-size displays. In *Proc. UIST'01*, (pp. 21-30). NY: ACM.
- Henderson, D. A. Jr., Card, S.K. (1987). Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5 (3), 211-243.
- Kandogan, E. and Shneiderman, B. (1997). Elastic Windows: evaluation of multi-window operations. In *Proc. CHI 97*. (pp. 250-257). NY:ACM.
- Kaptelinin, V. (2002). UMEA: User-monitoring environment for activities. In *Proc. UIST'02 Companion*, (pp. 31-32).
- MacIntyre, B., Mynatt, E., Volda, S., Hansen, K., Tullio, J., Corso, G. (2001). Support for multitasking and background awareness using interactive peripheral displays. In *Proc. UIST 2001*, (pp. 41-50).
- Malone, T. W. (1983). How do people organize their desks? Implications for the design of office automation systems, *ACM Transactions on Office Information Systems* 1 (1), 99-112.
- Myers, B. (1988). Window interfaces: A taxonomy of window manager user interfaces, *IEEE Computer Graphics and Applications*, 8 (5), 65-84.
- Mynatt, E., Igarashi, T., Edwards, W., and LaMarca, A. (1999). Flatland: new dimensions in office whiteboards. In *Proc. CHI'99*, (pp. 346-353).
- Patton, B.M. (1990). The history of memory arts. *Neurology*, 40, 346-352.
- Rekimoto, J. (1999). Time-machine computing: A time-centric approach for the information environment. In *Proc UIST'99* (pp. 45-54).
- Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D., and van Dantzich, M. (1998). Data Mountain: Using spatial memory for document management. In *Proc. UIST'98* (pp. 153-162).
- Robertson, G. van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Ridsen, K., Thiel, D., Gorokhovskiy, V. (2000). The task gallery: a 3D window manager. In *Proc CHI'00* (pp. 494-501).
- Teitelman, W. (1986). Ten years of window system - A retrospective view. In Hopgood, F., Duce, D., Fielding, E., Robinson, K., & Williams, A. (Eds.). *Methodology of Window Management*. Berlin: Springer-Verlag.
- Wurnig, H. (1998). Design of a collaborative multi user desktop system for augmented reality, in *Proc. Central European Seminar on Computer Graphics*.
- XDesk Software (2003), About Virtual Desktop Managers, <http://www.virtual-desktop.info>.

Office Action Summary

Application No.

10/045,919

Applicant(s)

PATIL, RAJESH V.

Examiner

Syed J. Ali

Art Unit

2195

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 09 January 2002.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-25 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-25 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 24 June 2002 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: _____

DETAILED ACTION

1. Claims 1-25 are pending in this application.

Claim Objections

2. **Claim 11 is objected to because of the following informalities:**

- a. In line 1 of claim 11, "System" should read: "A system".

Appropriate correction is required.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. **Claims 1-25 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.**

5. As per claim 1, the claim language raises a question as to whether the claim is directed merely to an abstract idea that is not tied to a technological art, environment or machine which would result in a practical application producing a concrete, useful, and tangible result to form the basis of statutory subject matter under 35 U.S.C. 101. The claimed "method" should be modified to indicate that it is embodied in a manner as to be executable, e.g. "a computerized method" or "a computer-implemented method". Claims 2-10 are rejected for at least the same

Art Unit: 2195

reasons as their parent claim, as they fail to present any limitations that resolve the deficiencies of the claim from which they depend.

6. As per claim 11, the claimed system is non-statutory as it is not tangibly embodied, in that it fails to include any hardware as part of the system. The system could be implemented entirely in software. Claims 12-14 are rejected for at least the same reasons as presented for their parent claim, as they fail to present any limitations that resolve the deficiencies of the claim from which they depend.

7. As per claim 25, the claimed "computer program element" is non-statutory as it is not tangibly embodied. The "computer program element" could be implemented entirely in software.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

9. **Claims 1-25 are rejected under 35 U.S.C. 102(a) as being anticipated by Philippot ("Rearrange Your Taskbar Buttons").**

10. As per claim 1, Philippot teaches the invention as claimed, including a method for run time ordering open tasks, comprising the steps of:

providing for each open task a task tab sequentially ordered in a status line (pg. 1, Configuring ButtonBoogie); and

responsive to a user drag and drop of a first task tab, reordering the sequential order of said status line (pg. 1, Introduction; pg. 1, Getting Started).

11. As per claim 2, Philippot teaches the invention as claimed, including the method of claim 1, further comprising the steps of:

providing a linked list of tab elements, with a tab element associated with each said task tab and task content (pg. 2, Commandeering the Taskbar; pg. 3, Handling Tab Control Messages); and

said reordering step comprising the step of adjusting back pointers and next pointers of said tab elements to reposition said first tab element from a drag position to a drop position within said linked list (pg. 1, Getting Started).

12. As per claim 3, Philippot teaches the invention as claimed, including the method of claim 2, further comprising the step of:

displaying in an application content window task content corresponding to a current selected task tab (pg. 1, Introduction; pg. 1, Configuring ButtonBoogie; pg. 2, Commandeering the Taskbar).

Art Unit: 2195

13. As per claim 4, Philippot teaches the invention as claimed, including the method of claim 3, further comprising the step of:

responsive to user selection of a new task tab of displaying task content corresponding to said new task tab in said new task tab in said application content window (pg. 1, Configuring ButtonBoogie; pg. 2, Commandeering the Taskbar).

14. As per claims 5-6, Philippot teaches the invention as claimed, including the method of claim 4, further comprising the step of:

responsive to user selection of a first key combination, displaying task content corresponding to a next sequential task in said application content window and responsive to user selection of a second key combination, displaying task content corresponding to a previous sequential task in said application content window (pg. 1, Introduction; pg. 2, Commandeering the Taskbar, wherein ButtonBoogie is disclosed as compatible with Windows, which inherently allows switching between open tasks using Alt+Tab and Alt+Shift+Tab).

15. As per claims 7-10, Philippot teaches the invention as claimed, including the method of claim 2, said task being a document, web page, database, or spreadsheet (pg. 1, Configuring ButtonBoogie).

16. As per claim 11, Philippot teaches the invention as claimed, including a system for run time ordering open tasks, comprising:

a status bar (pg. 1, Configuring ButtonBoogie);

a plurality of task tabs presented in said status bar in sequential order, with a task tab for each open task within an open application (pg. 1, Configuring ButtonBoogie);

a content window for displaying task content associated with a current selected task tab (pg. 1, Introduction; pg. 1, Configuring ButtonBoogie; pg. 2, Commandeering the Taskbar);

a linked list of tab elements, with a tab element associated with each task tab and task content (pg. 2, Commandeering the Taskbar; pg. 3, Handling Tab Control Messages); and

a task tab order component responsive to user drag and drop of a given task tab for moving said given task tab from a drag position in said status bar to a drop position in said status bar (pg. 1, Introduction; pg. 1, Getting Started).

17. As per claim 12, Philippot teaches the invention as claimed, including the system of claim 11, further comprising:

a mouse device operable by a user for selecting a current task tab (pg. 1, Introduction; pg. 1, Getting Started; pg. 3, Handling Tab Control Messages).

18. As per claim 13, Philippot teaches the invention as claimed, including the system of claim 12, further comprising:

a first key operable by said user for incrementing said selected task tab to a next tab in said sequential order and a second key operable by said user for decrementing said selected task tab to a previous tab in said sequential order (pg. 1, Introduction; pg. 2, Commandeering the Taskbar, wherein ButtonBoogie is disclosed as compatible with Windows, which inherently allows switching between open tasks using Alt+Tab and Alt+Shift+Tab).

19. As per claim 14, Philippot teaches the invention as claimed, including the system of claim 11, said task tab order component being operable to reorder said linked list of tab elements responsive to said user drag and drop (pg. 1, Introduction; pg. 1, Getting Started).

20. As per claims 15-24, Philippot teaches the invention as claimed, including a program storage device readable by a machine, tangibly embodying a program of instructions executable by a machine to perform the method of claims 1-10, respectively (pg. 1, Introduction).

21. As per claim 25, Philippot teaches the invention as claimed, including a computer program product or computer program element for performing the method of claim 1 (pg. 1, Introduction).

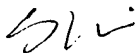
Conclusion


22. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Syed J. Ali whose telephone number is (571) 272-3769. The examiner can normally be reached on Mon-Fri 8-5:30, 2nd Friday off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai T. An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Art Unit: 2195

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).


Syed Ali
July 15, 2005


MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Notice of References Cited	Application/Control No. 10/045,919		Applicant(s)/Patent Under Reexamination PATIL, RAJESH V.	
	Examiner Syed J. Ali		Art Unit 2195	Page 1 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,334,157	12-2001	Oppermann et al.	719/310
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)			
	U	Lin, Mike. "Taskbar Commander" (1999). ✓			
	V	Philippot, Patrick. "Rearrange Your Taskbar Buttons", PCMag.com (December 2001).			
	W	Chang, Alexander. "A Better Windows", (2002).			
	X	Smith et al. "GroupBar: The TaskBar Evolved." Proceedings of OZCHI 2003 (2003).			

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Notice of References Cited	Application/Control No. 10/045,919	Applicant(s)/Patent Under Reexamination PATIL, RAJESH V.	
	Examiner Syed J. Ali	Art Unit 2195	Page 2 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Fotinis, Elias. "TaskArrange v1.1.1". (2005).
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

✧ **How to re-order taskbar buttons?**All 2 messages in topic - [view as tree](#)**Matt Warner** Sep 23 1999, 3:00 am [show options](#)

I am trying to find a way of changing the order in which applications appear on the taskbar. This may seem like a pointless thing to want to do, but a lot of people in my office have a lot of things in their startup folder. This means that the applications are usually in the same order.

However, when I use their machines to do general troubleshooting, I close some apps down and then when they reopen them they get confused because they are used to having, for example, OE on the left.

It's not a critical thing but I would like to find some easy way to

Matthew Ellis Sep 24 1999, 3:00 am [show options](#)

Matt Warner <train...@lichfield.anglican.org> wrote in message

[news:#wmt7AbB\\$GA.252@cppssbbsa05...](mailto:news:#wmt7AbB$GA.252@cppssbbsa05...)

> I am trying to find a way of changing the order in which applications appear
> on the taskbar. This may seem like a pointless thing to want to do, but a
> lot of people in my office have a lot of things in their startup folder.
> This means that the applications are usually in the same order.

> However, when I use their machines to do general troubleshooting, I close
> some apps down and then when they reopen them they get confused because
> they
> are used to having, for example, OE on the left.

> It's not a critical thing but I would like to find some easy way to

know exactly this problem. I like having (in order) explorer, OE, and then developer studio. I get a little lost if they're not in their right places :)

Anyway, what you want is ITaskBarList. Using DeleteTab and AddTab will allow you to re-order the task bar. Unfortunately, this only works with the IE4 Desktop Update. If you don't have IE4, you can't do it.

Some nice chappy has written a simple program to do what you want. Check it out - you want

Matthew
m.t.el...@swansea.ac.uk

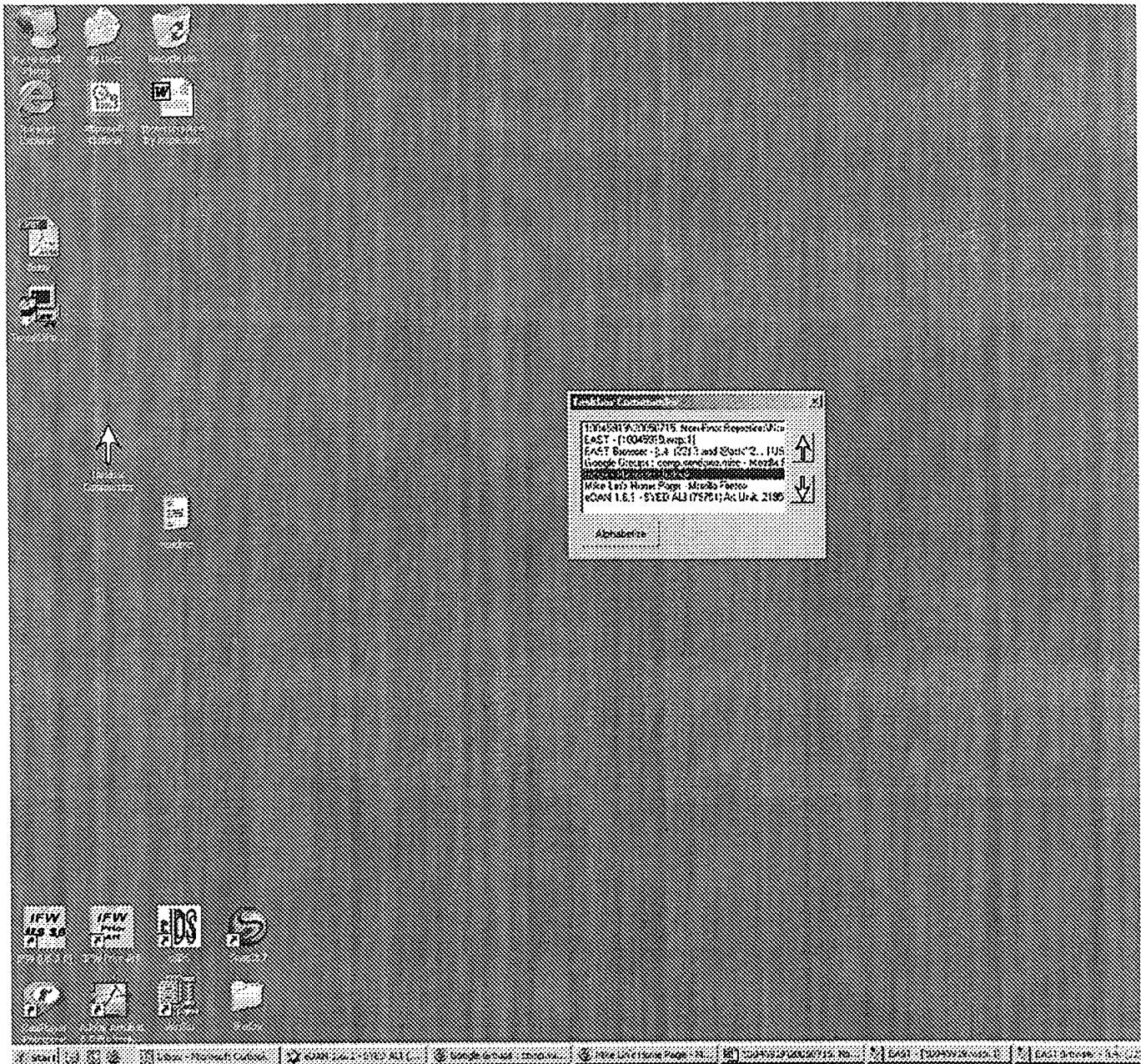
readme

Taskbar Commander

by Mike Lin <m1in@m1in.net>

Taskbar Commander is a quick utility for the eccentric windows user - it allows you to reorder the buttons on your taskbar and arrange them as you like.

Usage is pretty simple. Upon startup, TC will alphabetize your taskbar buttons. You can then use the pretty straightforward controls to move the buttons around.



Download FreeTrial

Fix Computer Errors - Easy
Repair all errors - Free Download! Your PC
will run better- Guaranteed

Google

Search

/ [home](#) /[Web Scripts Directory](#)[Submit Software](#)[Advertise](#)**Categories**

Utilities
Business
Desktop
MP3 & Audio
Multimedia Tools
Internet Programs
Games
Web Authoring
Web Scripts
Development
Educational
Hobbies
Miscellaneous
Resource Links
Marketplace
Products
Books
Cleaning & Office Supplies

Authors

Submit Software
Offers to Developers
Affiliate Programs

Linux Resources

Linux Resources Home
Linux Guides
HOWTO's
Linux Disctionary

[Parental Controls](#)
[FREE Content Scanner -](#)
[Detect Porn - Content](#)
[Cleaner - Delete Porn](#)

[Site Map](#)

Taskbar Commander 1

Rearrange Windows Taskbar Icons

O.S. : Win98, WinME, WinNT
4.x, WinXP, Windows2000
License: Shareware \$5

Min Requirements : .Net Redistributors

Limitations :

Filesize: 0.19 MB

Screen Shot : [Click Here](#)

Program Website : [OnlineToolsTeam](#)

Ads by Goooooogle

Windows Startup

Clean registry to improve start-up time and applications performance.
[www.pctools.com](#)

Windows Errors Free Help

Free Windows Errors Diagnostic Tool Free Downloads Fix Windows Now.
[www.windows-repairs.com](#)

Fix Windows Errors - Free

2005 Most-Advanced Error Remover. Fix Your Computer - Free Download!
[ErrorNuker.com](#)

You will no longer have to settle for the windows default taskbar location. By using Taskbar Commander you can rearrange your programs automatically within the windows taskbar to any way you like and have it remember your favorite locations. Taskbar Commander can automatically move a new app to replace a closed app that was within your specified applications range. You can also select any number of applications and move them within your taskbar with a drag drop, by clicking on the arrows or by holding down the ctrl button and using the up and down arrow keys. Taskbar Commander requires .Net redistributable to run. For more information on .Net and/or a complete list of features and screen shots visit our website at [www.OnlineToolsTeam.com](#)

Download Taskbar Commander: [Download Site 1](#)
[Download Site 2](#)

Recommended Related Software Titles

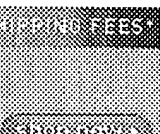
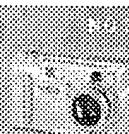
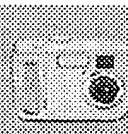
Iconic Tray 1.21 - Minimize any window to a special new tray...

Brizo Trevi Bidet - 6316838 -

This page is [mirrored](#) on [Download FreeTrial](#)



Digital made simple
with Click & Learn articles
Great gifts for back-to-school



Rearrange Your Taskbar Buttons

ARTICLE DATE: 12.26.01

By Patrick Philpott

When you're moving back and forth between two applications, having their taskbar buttons next to each other is convenient. But unless you opened the applications one right after the other, their buttons will not be adjacent. In another case, you may have several browser windows open, and the one on the left is associated with the right-most taskbar button. This can be confusing! ButtonBoogie solves the problem by letting you rearrange taskbar buttons using drag-and-drop, which is much easier than closing and reopening the applications. Also, by dragging or clicking taskbar buttons with certain keys pressed, you can set the default window size and placement of applications on the Desktop. ButtonBoogie also adds some functionality to the taskbar. It allows you to easily (and optionally, automatically) set particular application windows to a predefined size, state, and position.

ButtonBoogie is fully supported under Windows 95, 98, Me, NT 4.0, and 2000. The program doesn't work under Windows XP, because Windows XP implements the taskbar in a completely different way from other Windows platforms.

Getting Started

To install ButtonBoogie, run the supplied installation program and follow the instructions. Two shortcuts will be added to your Start | Programs menu: *Configure ButtonBoogie* and *Start ButtonBoogie*. During installation, you'll be asked whether you want a *Start ButtonBoogie* shortcut to be added to your Startup folder. Check this option if you want ButtonBoogie to be active as soon as your Windows session starts. To remove ButtonBoogie from your system, use the Control Panel applet Add/Remove Programs.

ButtonBoogie has two modes. The *Start ButtonBoogie* shortcut uses quiet mode (the */q* command line switch). This silently and invisibly loads ButtonBoogie on your computer. The *Configure ButtonBoogie* shortcut runs without the */q* command line switch, launches ButtonBoogie, and invokes a dialog box that allows you to set ButtonBoogie's options.

Note that once ButtonBoogie is loaded onto your system, it can't be unloaded without logging off and on again or rebooting. This means that if you uninstall ButtonBoogie after having launched it, you will have to reboot your computer. You can disable ButtonBoogie without uninstalling it using the *Configuration* dialog.

Once ButtonBoogie is running, you can reposition a taskbar button by dragging it onto the button whose position you want to take. If the dragged button is moved from the right to the left, the replaced button and all buttons on its right will be shifted to the right. If the dragged button is moved from the left to the right, the replaced button and all buttons on its left will be shifted to the left.

Configuring ButtonBoogie

Launch the *Configuration dialog* by clicking on the shortcut *Configure ButtonBoogie*. This invokes a dialog box (Figure 1) that allows you to modify four options: *Enable ButtonBoogie*, *Allow automatic button relocation*, *Allow automatic resizing*, and *Display tray icon*.

To temporarily disable all ButtonBoogie functions, clear the *Enable ButtonBoogie* check box.

When you launch a program or when you open a new window, a new taskbar button is usually created. If *Allow automatic relocation* is checked, ButtonBoogie will try to relocate a newly created taskbar button to the position where you last placed it.

When you drop a taskbar button onto a new location, ButtonBoogie remembers this location. When a taskbar button is already located where you want it to be, you must move that button to another location and then move it back to its original location for the location to be recorded.

The *Allow automatic resizing* option is related to ButtonBoogie's moving and sizing features. As explained in the next section, ButtonBoogie allows you to record a window's size, placement, and state (minimized, maximized, or normal) as the default for that application. If you check the *Allow automatic resizing* option, a newly created application window will automatically be set to its default values (assuming you have set default values for this application).

The option *Display tray icon* lets you tell ButtonBoogie whether to display its icon in the system tray. Right-clicking this icon gives you direct access to the configuration options described here (see Figure 2). It also gives you easy access to the Help file, which lists the special hotkeys used to set a window's size and placement.

Moving and Sizing Windows

ButtonBoogie offers a very simple way to set your application windows' sizes, states (maximized, minimized, or normal) and positions to predefined values. You can assign a default size, state and position to an application window and reset the window to these values manually or automatically when the window is launched. You accomplish this by holding down special keys while dragging, dropping, or clicking on the taskbar buttons.

A *Shift-left-click* on a taskbar button stores the current size, state, and position of the target window as default values for this window. The mouse cursor will briefly change to the BB cursor while storing the data, and the default system beep will sound.

An *Alt-left-click* on a taskbar button resets the window to its default size, state, and position if those values have been stored.

A *Shift-drag* (dragging a source taskbar button onto a target taskbar button) stores the current size, state, and position of the source window as defaults for the target window. The default system beep will play and the BB cursor will be maintained for about 1 second.

An *Alt-drag* (Drag a source taskbar button onto a target taskbar button) will temporarily set the current size, state, and position of the target window to the size, state, and position of the source window. In this case, nothing is stored; the change is temporary.

Note that the size and position of the source window in its normal state are applied to the target window, even when the source window is maximized or minimized. If the target window is not resizable, only the new position will be taken into account.

An *Alt-Shift-left-click* on a taskbar button deletes any information relating to the corresponding window, including the preferred location of the taskbar button.

Note that some applications handle their main windows in ways that ButtonBoogie can't correctly track. In such cases, the moving and sizing functions and the automatic button relocation feature may not work.

Ctrl-left-click lets you select multiple taskbar buttons and apply commands like *Close* and *Cascade* to the group by right-clicking one of the selected buttons. This is a built-in Explorer feature.

Inside ButtonBoogie

ButtonBoogie is made up of two modules. BtnHook.dll is a DLL that is loaded into the Windows Explorer process and modifies the way taskbar buttons are handled. ButtonBoogie.exe serves two purposes: In quiet mode (/q command line switch), it loads BtnHook.dll into Explorer; in non-quiet mode, it displays a small options dialog. The most interesting part of ButtonBoogie is the hook DLL. ButtonBoogie.exe is just a simple dialog that modifies options in the program's INI file. It also communicates with the hook DLL by means of private messages-more on this later. ButtonBoogie was developed with Visual C++ using plain C/SDK code-no MFC (Microsoft Foundation Classes) and no ATL (Active Template Library). This utility must have the smallest possible footprint, because it loads in Explorer's memory space. In the sections that follow, I will focus on the problems I discovered during the development of this program, which are mostly related to the way Explorer handles the tab control displaying the "pseudo" taskbar buttons.

Commandeering the Taskbar

Before explaining how ButtonBoogie works, I must explain how the taskbar buttons work. Although they look and behave like normal buttons, the set of taskbar buttons is actually a tab control managed by Explorer.exe. You can observe this by using the Spy++ tool provided with Microsoft Visual C++. Each button is a tab belonging to the tab control SysTabControl32. (This is not true in Windows XP, where the tab control is replaced with a toolbar control.) Also note that the tab control has the TCS_OWNERDRAWFIXED style. This suggests that Explorer also subclasses this tab control, so conflicts may arise.

Once you know that the taskbar buttons are actually tabs in a tab control, you have to come up with a way to interact with this control a way to subclass it. In order to subclass a control in another process, you must get your subclass window procedure loaded within the other process's memory context. I didn't want to have a permanently loaded program subclassing this control. This brute force approach is too intrusive for a simple install-and-forget utility, and it consumes more system resources than warranted for such a simple task. Two other methods are available for dynamically adding code to Explorer: creating a BHO (Browser Helper Object) or using the DLL injection method. Using a BHO had some drawbacks: providing a minimal user interface to the utility via a BHO is not easy, and the BHO would have to support a whole set of COM interfaces that would be useless given the purpose of this utility.

I finally decided to use the DLL injection method. There's a Shell API function named SHLoadInProc that is supported in all Windows environments except for Windows CE. This API function takes a single parameter that is technically the CLSID of a COM object, and loads the DLL defining that COM object into Explorer's address space. In fact, as long as the DLL provides the required entry points DllGetClassObject, DllCanUnloadNow, DllRegisterServer, and DllUnregisterServer, the DLL doesn't have to contain a. The DLL just needs to be registered as if it contained a COM object. When you call SHLoadInProc and pass it the pseudo COM object's CLSID, this function loads the corresponding DLL into Explorer's address space. The usual notifications are sent to this DLL through DllMain, and you can do whatever you want when receiving these notifications.

Our pseudo COM object uses the DLL_PROCESS_ATTACH notification to find and subclass the SysTabControl32 window. This window is easy to locate. Starting from the taskbar itself, you can use

FindWindow to walk down the window hierarchy this way:

Shell_TrayWnd | RebarWindow32 | MStaskSwWClass | SysTabControl32

Note that the RebarWindow32 control is there only if the WDU (Windows Desktop Update) is installed.

Once the tab control window has been found, you can subclass it by using the SubclassWindow macro. Then we just have to wait for the messages sent to the tab control. We can intercept these messages in our NewSysTabControlProc window procedure and handle them according to our needs. This is where I encountered a few surprises.

Handling Tab Control Messages

To track drag-and-drop actions on the taskbar, we would just have to intercept the WM_LBUTTONDOWN, WM_LBUTTONUP and WM_MOUSEMOVE messages. We can easily trigger a drag operation in the WM_LBUTTONDOWN handler, track the hovered window (button) in WM_MOUSEMOVE, and determine the final target in the WM_LBUTTONUP handler. The TabCtrl_xxxx family of macros gives us enough power to manage the tabs in the tab control. Because there is no available function for moving a button directly, we have to first delete the source button after saving the relevant information and reinsert a new tab at the target location. This is done in the MoveSysControlTab function.

The first annoying problem that I encountered is that the tab control window procedure is not re-entrant. Most of the time, you can't use the TabCtrl_xxxx functions while handling a tab control message. This either crashes Explorer or gives erratic results. The workaround is to use private messages (WM_USER + n) that are self-posted to the tab control when our hook needs to trigger a special action like moving a button or resizing a window. You'll find the definitions of these private messages in *common\b.b.h*.

Also, I noticed that using a global, statically allocated TCITEM structure to retrieve information about taskbar buttons caused the same symptoms to appear: Explorer crashed and erratic results were returned by the TabCtrl_xxxx functions. Using a dynamically allocated structure solved the problem, though the reasons why are not completely clear. When the structure is static, it's located in the BtnHook.dll data segment. When dynamically allocated, it's located on Explorer's process heap. Explorer apparently does something special with the control that is not compatible with static allocation.

Detecting a New Taskbar Button

The next problem was how to detect new buttons. ButtonBoogie needed to do this to implement the automatic relocation and automatic resizing features. I naively thought that a new button would be added to the taskbar using the standard TCM_INSERTITEM message sent by Explorer to the taskbar. Obviously, Explorer has its own way of doing things, probably because the whole taskbar is owner-drawn. In any case, the insertion of new buttons seems to be triggered by sending private, undocumented messages to the SysTabControl32 window.

Because I didn't want to interfere too much with Explorer's own tab control handler, I decided to use a safe method for detecting new buttons. A timer is triggered every 3 seconds, and each time, ButtonBoogie checks whether the number of taskbar buttons has increased. If it has, ButtonBoogie considers the right-most button to be the new button and tries to relocate the button and to resize the corresponding window if the automatic resizing option is on. This is done in the ReOrder and SetSizeAndPos functions.

ButtonBoogie then uses Explorer's unusual behavior to its advantage. Because Explorer never inserts a new button using the TCM_INSERTITEM message, we know that when this message is received, it's because we have just moved a tab (deleted and then reinserted that tab). This makes operations like collecting the new location of a button much easier.

Keeping Track of Windows

ButtonBoogie allows users to automatically relocate taskbar buttons and automatically resize and relocate the windows attached to a given button, so we need to store this information somewhere. An INI file is the ideal location, but each window must be identified in a reproducible way. How to do this may seem obvious, but it is not.

Once you have identified a particular taskbar button, getting to the attached window is easy, though undocumented. A good debugger will quickly show you that the IPParam field of the TCITEM structure stores the window handle of the window associated with the button. Given the window handle, retrieving the window caption or the window class name is easy. Likewise, because you have the source window handle at your disposal, you can easily retrieve placement information using the standard API functions GetWindowPlacement and SetWindowPlacement during operations like *Shift-click* or *Alt-click*.

I first used the window caption to identify a particular window. But this gave poor results, because many applications append or prepend a document name to the window caption. If this happened, ButtonBoogie would have a hard time retrieving the information about a particular application's main window from Buttonboogie.ini. If the window caption were different each time, no match would be found.

So I decided to use the window class name instead. This gave much better results, though not with every application. Some MFC-based applications (like the Netscape Communicator package) have dynamically computed and very cryptic class names. In this case, ButtonBoogie uses the window caption, which makes it susceptible to the previously described problem.

I considered using the process name to identify the window, but this would have generated even more annoying problems. Some processes (like Explorer itself) manage very different tasks in very different windows, and using the same identifier for all these windows would give catastrophic results.

Sometimes, keeping track of an application window is virtually impossible. In such cases, only the button-moving feature will be available for the application's windows.

Communicating with the Configuration Dialog

The ButtonBoogie tray icon is handled by the hook DLL, because we need a permanently available window to receive the notification messages sent by the tray icon. The configuration dialog box (Buttonboogie.exe) and our hook are not running in the same process, but they have to be able to communicate. After all, users can change program options either from the dialog box or the ButtonBoogie tray icon menu.

ButtonBoogie solves this problem by using private messages sent both ways. The program-defined WM_RESETBTRAY message tells the hook DLL that it should re-read the *Display tray icon* option value and display or hide the tray icon accordingly. Likewise, the WM_RESETBBDIALOG message tells the configuration dialog that it should reread the user options and set the dialog's check boxes accordingly.

Conclusion

The SHLoadInProc API function is a very handy function if you need to modify the standard behavior of Explorer but want to avoid the hassle of implementing a full-blown BHO. You should be aware, however, that Microsoft's developers don't always program the way they tell us to program. The tab control used to display the taskbar buttons is handled in a very special way, so we shouldn't be too intrusive when modifying its behavior.

The Windows XP case is unusual not only because it uses another kind of control to handle taskbar buttons, but also because it implements a button grouping feature. This creates additional difficulties for an application like ButtonBoogie. I'm working on this problem currently.

ButtonBoogie seems like a simple program on the surface, but under the hood, it is a complex utility that employs some sophisticated tricks to accomplish its tasks.

Patrick Philpott, the author of ButtonBoogie, is a developer based in Draveil, France. Sheryl Canter is the editor of the Utilities column and a contributing editor of PC Magazine.

ButtonBoogie: Download It Here

ARTICLE DATE: 12.26.01

By Patrick Philpott

First Published on the PC Magazine Extra Web site, December 26, 2001 (v20n22)

Utility: ButtonBoogie, Version 1.1

Platforms: Windows 95, Windows 98, Windows ME, Windows NT 4, Windows 2000

License Information:

PC Magazine programs are copyrighted and cannot be distributed, whether modified or unmodified. Use is subject to the terms and conditions of the license agreement distributed with the programs.

Description:

When you're moving back and forth between two applications, it's convenient to have their taskbar buttons next to each other. But unless you opened the applications one right after the other, their taskbar buttons will not be adjacent. Or maybe you have several browser windows open and the one on the left is associated with the right-most taskbar button. This can be confusing! ButtonBoogie solves this problem by letting you rearrange taskbar buttons using drag-and-drop. This is much easier than closing and reopening the applications. Also, by dragging or clicking taskbar buttons with certain keys pressed, you can set the default window size and placement of applications on the desktop. ButtonBoogie was written by Patrick Philpott, and first appeared in PC Magazine December 26, 2001 (v20n22). Source code is included.

Copyright (c) 2005 Ziff Davis Media Inc. All Rights Reserved.

TaskArrange v1.1.1

Copyright © 2004-2005 by Elias Fotinis
2005.04.15 - Freeware

[Purpose](#) - [What's New](#) - [Usage](#) - [Options](#) - [Language Packs](#) - [Known Issues](#) - [Installation](#) - [Uninstall](#) - [Requirements](#) - [History](#) - [Contact](#) - [Legal](#)

Purpose

TaskArrange is a simple utility that lets you rearrange the buttons of the Windows taskbar.

Sometimes we open our programs in a specific sequence, to keep their taskbar buttons in a desired order. But what happens if a program crashes or we close it, and then we open it again? That's right - its task button ends up last in the taskbar. Windows itself does not allow us to move the task buttons around, so we are stuck with two options: either accept the new order, or close everything and reopen them in the preferred order. TaskArrange brings an end to this annoyance, by letting us do exactly what we want: *reorder the task buttons*.

What's New

- The correct order of the buttons can now be detected in NT4.0.

Usage

When you start TaskArrange, it displays a **list** of the taskbar windows. You can rearrange this list in several ways:

- By using the buttons at the bottom of the list.
- By dragging the list items with the mouse.
- By pressing Ctrl+Up (moves up), Ctrl+Down (moves down), Ctrl+Home (moves to top), and Ctrl+End (moves to bottom).

When the desired order has been set, press the **OK** button to rearrange the actual buttons in the taskbar and quit TaskArrange, or press **Apply** to perform the action and keep the program running.

Refresh will update the list with the current buttons, in case new windows were opened while TaskArrange was running.

WinXP has an option to *group similar taskbar buttons*. If this is in effect, TaskArrange will display the grouped buttons indented under the name of their application. You can move the whole group by selecting the application. You can also rearrange the buttons within the group, but you cannot move them outside the group or insert a button from a different group. You can also move the whole group by selecting the application itself. If the grouped buttons in the taskbar have been replaced by their application's button, their titles will be grayed in TaskArrange. Otherwise, the application name will be grayed instead.

TaskArrange uses a special method to accurately identify the taskbar buttons. Should this method fail for some reason, TaskArrange will enter a **safe mode**, displaying a relevant message in its caption. In safe mode, the order of the taskbar buttons cannot be determined, so they will be listed randomly.

Furthermore, it is possible that some buttons will not be detected at all. However, even in safe mode, you can still rearrange the list and apply the changes.

Options

You can configure TaskArrange by clicking on **Options**. The available options are:

- **Remember window position**
Saves the position of the program window on exit and restores it on the next execution.
- **Exit with Escape**
Enables you to quit TaskArrange by hitting Escape.
- **Language**
Allows setting the preferred language used by the program's interface and help. See [Language Packs](#) for more details.

Language Packs

You can install language packs in order to use translated versions of the program's interface and help:

1. Download a language pack from the program's homepage.
2. Create a subfolder named **Lang** in the program's folder.
3. Extract the files from the language pack into the **Lang** folder.
4. Select the language in the **Options** dialog.

You can easily translate TaskArrange's interface and/or help, by downloading the translation pack from the program's webpage. You'll need a resource editor to edit the template DLL; you can use MSVS6/7 or the free [Resource Hacker](#), among others.

Known Issues

There are a few possible problems you might run into:

- The "*taskbar buttons grouping*" option in Windows XP makes it impossible to separate the grouped buttons. If you want complete control over the buttons' arrangement, you should turn this option off (right-click on the taskbar, select Properties and uncheck "Group similar taskbar buttons").

Installation

Due to its small size, TaskArrange does not come with an installer. Instead, you should extract the contents of the zip package into a folder of your preference and optionally create a shortcut for the program file.

In the downloaded package you'll find two versions of the program:

- **TaskArrange.exe**
This will only run on NT-based Windows, which include NT, 2000, XP, and 2003. It is the Unicode version and thus it will correctly display international characters. It cannot be run on Win95/98/Me.
- **TaskArrange(9x).exe**
This will run on any Windows version, but it cannot properly display Unicode characters on NT-

based systems. You should only use this version if you are running Win95, 98, or Me.

Uninstall

To uninstall TaskArrange, simply delete the files you extracted from the zip file. TaskArrange does not store any settings in the Registry or alter any system files.

Requirements

There are no specific requirements for running TaskArrange. It has been tested successfully in the following operating systems:

Win 95, 98, ME, 2000, XP

It is recommended (but not required) to have IE4+ installed, in order to better manipulate the taskbar buttons.

History

Version/Date	Notes
1.1.1 - 2005.04.15	The correct order of the buttons can now be detected in NT4.0.
1.1 - 2004.12.20	Better support for WinXP grouped buttons. Localization support. List keyboard shortcuts; OK button; per-user settings stored in INI file. Both Unicode and non-Unicode versions included. Bug fix: taskbar grouping in combination with button sliding effect no longer prevents button arrangement in WinXP.
1.0 - 2004.09.02	First public release.

Contact

Comments and suggestions are welcome. Just send an email at:
efotinis@pat.forthnet.gr

For future updates and more freeware, please visit my website at:
<http://users.forthnet.gr/pat/efotinis/>

Legal

This program is provided "as-is". The author cannot be held liable for any damages caused by the use or misuse of this program. You can redistribute this program in its unmodified form, but you must not charge any money for its usage. All trademarks mentioned in this file are the property of their respective owners.

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.